10/530,953

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 September 2002 (12.09.2002)

PCT

(10) International Publication Number
WO 02/071021 A1

(51) International Patent Classification⁷: G01L 19/00

(21) International Application Number: PCT/US02/06710

(22) International Filing Date: 28 February 2002 (28.02.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/798,794 2 March 2001 (02.03.2001) US

(63) Related by continuation (CON) or continuation-in-part (CIP) to earlier application:
US 09/798,794 (CIP)
Filed on 2 March 2001 (02.03.2001)

(71) Applicant *(for all designated States except US)*: **FIRST INTERNATIONAL DIGITAL, INC.** [US/US]; 135 West Central Road, Schaumburg, IL 60195 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **QIAN, Ziqiang** [CN/US]; 5131 N. Tamanack Dr., Barrington, IL 60010 (US). **JOHNSON, Timothy** [CN/US]; 175 Aberdeen, Algonquin, IL 60102 (US).

(74) Agent: HAMMAN & BENN; 10 South LaSalle Street, Suite 3300, Chicago, IL 60603 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

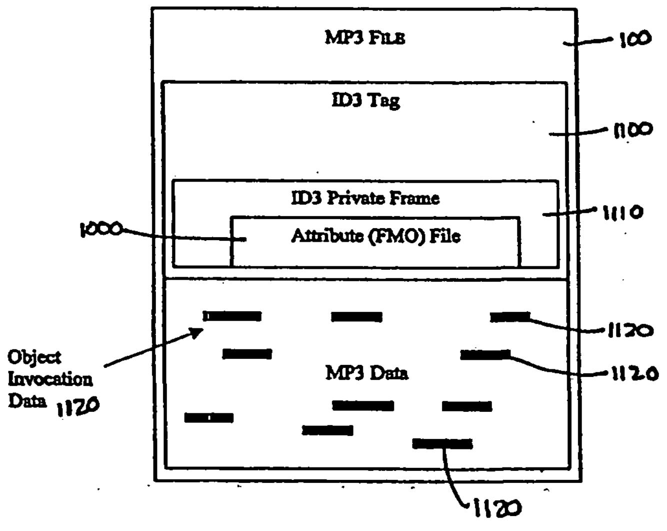
(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

BEST AVAILABLE COPY

(54) Title: METHOD AND SYSTEM FOR ENCODING AND DECODING SYNCHRONIZED DATA WITHIN A MEDIA SEQUENCE



(57) Abstract: A method encodes a media sequence (100) with at least one object (200) defined as an applet object (1010) and a corresponding data object (1020). The object (200) is inserted into at least one attribute file (1000). A media sequence (100) is provided with a media file. The attribute file (1000) is integrated into the media file and a link (1120) is provided to invoke the objects is inserted.

WO 02/071021 A1

METHOD AND SYSTEM FOR ENCODING AND DECODING A MEDIA SEQUENCE

Field of the Invention

[1] In general, the invention relates to the field of digital audio recording. More specifically, the invention relates to audio sequences within a digital audio recording and in particular to the encoding and decoding of synchronized data within an audio sequence.

Background of the Invention

[2] With the rise in popularity of karaoke as an entertainment means, more and more songs are put in karaoke format. As a result, the need to transport and store these ever-growing musical libraries has become paramount. In some instances, digitized data representing the music and the lyrics has been compressed using standard digital compression techniques. For example, one popular current digital compression technique employs the standard compression algorithm known as Musical Instrument Digital Interface (MIDI). U.S. Patent No. 5,648,628 discloses a device that combines music and lyrics for the purpose of karaoke. The device in the '628 Patent uses the standard MIDI format with a changeable cartridge which stores the MIDI files. MIDI compatible devices however, require a physical size deemed obese by the consumer demand for smaller hand held devices.

[3] The International Organization for Standardization (ISO/IEC) has produced a number of generally known compression standards for the coding of motion pictures and audio data. This standard is generally referred to as the MPEG (Motion Picture Experts Group) standard. The MPEG standard is further defined in a number of documents: ISO/IEC 11172 (which defines the MPEG 1 standard) and ISO/IEC 13818 (which defines the MPEG 2 standard), both of which are incorporated herein by reference. Another, non-standard compression algorithm, which is based on MPEG 1 and MPEG 2 standards, is referred to as MPEG 2.5. These three MPEG versions (MPEG 1, MPEG 2, MPEG 2.5) are often referred to as "MPEG 1/2." U.S. Patent

No. 5,856,973 discloses a method for communicating private application data along with audio and video data from a source point to a destination point using the MPEG 2 format, designed for the broadcasting of television quality sample rates.

[4] The MPEG audio formats are further broken into a number of "layers." In general, the higher an MPEG audio format and the higher the layer is labeled, the more complexity is involved. The third layer, Layer III for the above mentioned MPEG audio formats is commonly known as the MP3, which has established itself as an emerging popular compression format for encoding audio data in an effort to produce near-CD quality results.

[5] In order to compensate for consumer preferences, smaller digital music players using the MP3 compression standard have been produced with built in displays to provide the audio, text, and graphics needed for Karaoke. These devices have become even more popular with the availability of hundreds of thousands of song titles now in the MP3 format. With such a consumer demand, large numbers of portable digital music players have become available, with even more soon to be released to the consumer market. Although these portable digital music players share one common feature, the ability to play audio of various formats, they are virtually non-compatible with each other because most have proprietary interfaces, custom operating systems (OS), and non-standard display systems.

[6] As the relatively new portable digital music player market becomes increasingly competitive, companies will struggle to find novel features in an effort to obtain or maintain differentiation among each others products. In addition, as devices such as personal data assistants (PDA's) and cellular telephones begin to integrate digital audio technology, makers of portable digital media players will be forced to adopt technologies and features associated with products in those markets. The line between general purpose PDA's, cellular telephones, and media players will soon become increasingly blurred for some market segments.

[7] Interoperability among these various devices however can only be possible with the definition and adoption of standards. Currently, there is no unified means of distributing non-audio, interactive content to and from portable music players.

Without a unified means or "standard" for providing non-audio data, innovation among manufacturers of general-purpose PDA's, cellular telephones, and media players will stagnate.

[8] Therefore, it would be desirable to have a method and system for encoding and decoding interactive text, graphics, and sound in a manor that improves upon the above-mentioned situations and prior art. Ideally, such a technology would be adaptable for consumer devises utilizing varying compression standards, file formats, and CODECs as are known in the art.

Brief Description of the Drawings

[9] **FIG. 1** is a block diagram illustrating an MPEG audio bit stream and its components as described in the MPEG audio specification standard and in accordance with the present invention;

[10] **FIG. 2** is a structure diagram of an object;

[11] **FIG. 3** is a structure diagram of a class;

[12] **FIG. 4** is a structure diagram of a function;

[13] **FIG. 5** is a structure diagram of a parameter;

[14] **FIG. 6** is a block diagram of a data frame structure within the MPEG audio bit stream of **FIG. 1**, in accordance with the present invention;

[15] **FIG. 7** is a block diagram of a data chunk component within the data frame structure of **FIG. 6**, in accordance with the present invention;

[16] **FIG. 8** is a block diagram of object mode code syntax for one embodiment of the data chunk component of **FIG. 7**, in accordance with the present invention;

[17] **FIG. 9** is a block diagram of the bit position and identification of object flags, in accordance with the present invention;

[18] **FIG. 10** is a block diagram of an attribute file; and

[19] **FIG. 11** is a block diagram of an MP3 file showing in one embodiment the placement of an attribute file within an ID3 tag.

Detailed Description of The Invention

[20] In the present invention, a preferred embodiment for encoding an audio sequence with synchronized data takes place according to the MPEG audio standard, as described above. The present invention is further applicable to any frame-based audio format, such as but not limited to MPEG 1, Layer III, as described in ISO/IEC 11172-3:1993 TC1:1996, *Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbits/s, Part 3, Audio;* MPEG 2, Layer III, as described in ISO/IEC 13818-3:1998, *Information Technology - Generic Coding of Moving Pictures and Associated Audio, Part 3, Audio*; MPEG 2.5, Layer III; and Advanced Audio Coding ("AAC") as described in ISO/IEC 13818-7:1997, TC1:1998, *Information Technology - Generic Coding of Moving Pictures and Associated Audio, Part 7, Advanced Audio Coding.* As such when used herein the term MP3 may refer to an audio sequence formatted in any of the above mentioned frame-based audio formats.

[21] Illustrated in **FIG. 1**, an MPEG audio file (bit stream), such as MP3, and its components **100** are associated with one embodiment of the invention. Alternative embodiments may use any media file containing unused bit portions. A media file may be defined as any file containing audio, video, graphics, or text; or promotes user interactivity. An MP3 file can be built up from a succession of small parts called frames **110**. A frame **110** may be comprised of a data block and a data block header **120** and audio information **130**, wherein the audio information are segments of an audio signal or audio file. The MP3 frames **110** are organized in the manner illustrated in **FIG. 1**, where the header **120** consists of 32 bits, and the CRC (Cyclic Redundancy Code) **140** may have either 0 or 16 bits depending on if error detection has been applied to the bit stream **100**. Side information **160** can occupy 136 bits for a single channel frame or 256 bits for a dual channel frame. The side information **160** can be divided into a main data begin **145**, private bits **150**, and rest of the data **155** segments. Samples **170** known in the art to contain Huffman coded audio signals, along with ancillary data **180**, may use the rest of the available frame **110** bits. In MP3 files **100**, frames **110** are often dependent of each other due to the possible use of

a "bit reservoir", which is a kind of buffer known in the art. It should also be noted that the present invention would find similar applicability in a video file or other formats of audio files.

[22] The size of a complete frame **110** can be calculated from its bit rate, sampling rate and padding status, which is defined in the header **120**. The formula for computing frame size is

$$FrameSize = \begin{cases} 144xBitRatelSamplingRate & \text{if the padding bit is cleared} \\ 144 \times BitRate / SamplingRate + 1 & \text{if the padding bit is set} \end{cases}$$

where the unit for FrameSize is byte. For example, to compress a stereo audio with 44.1 kHz sampling rate to a bit rate of 128 kbit/s, the *FrameSize* can be either 417 or 418 bytes, depending on the padding bit. The size of both samples **170** and ancillary data **180** may be determined from the header **120** and side information **160**.

[23] For one embodiment of the invention, synchronized lyrics or text and control information, which can be displayed or invoked while playing an MP3 file, needs to be embedded within the MP3 file. One way to embed the data is to use the ancillary data **180** component of a frame **110** but alternative embodiments may use different data locations. By reserving, a predefined amount of bits, such as 16-bits, from each ancillary data component **180** within each MP3 frame **110** for embedding data, a new file named MP3K can be generated from the regular MP3 file, without changing the MP3 bit stream **100** standard. The MP3K file is a media file name and may be used with embodiments of any media format or standard processed by an embodiment of the invention. One embodiment of the invention provides that the complete media and data information be contained in a bit stream called a media sequence, which may consist of one or more media files. The data information is embedded into the bit stream prior to encoding.

[24] Another embodiment of the invention may use an object-oriented design approach to organize the embedded data within the ancillary data **180** components.

An object-oriented design can simplify the updating, structure, and maintenance of embedded data.

[25] An object can be a subset of predefined functions and data. The object can either be an applet object or a data object. For simplicity purposes only, an applet object can be thought of as a program that can be used to do a very specific task, such as displaying the phrase of a song on the display. A data object contains all of the information needed for an applet object to perform its task. In terms of the synchronized lyric application, this is the actual lyrical data. As such, lyrics, text, system control, and/or display information may be encapsulated by objects. The structure of the objects, or the preferred structure for the MP3K objects, is shown below. However, alternative embodiments may use different structures.

[26] Referring now to **FIG. 2**, each object **200** can be uniquely identified by a 32-bit group number (GN) **210**. The number of functions **220** defined by the object **200** is specified in the object header, discussed in greater detail below. As the object **200** is loaded into the processing device (MP3 player, PC computer, cell phone, or other embodiment) the objects are registered. During registration, a table may be constructed with the entry point of each object in memory so that when referenced or invoked, each object **200** can be easily located and executed. The processing devices for this embodiment of the invention, typically consist of a player and player programs, such as are found in a MP3 player. Alternative compression-oriented audio processing devices or media programs capable of processing the MP3K (or alternative format) data may be used. Additionally, an encoded media sequence may be transferred to a device medium. A device medium may include, but is not limited to, wireless transmission, compact disc, network databases, and static memory.

[27] The object **200** also include two arrays a data structure array **240** and a function pointer array **250** that essentially tell the processing devices where data objects and applet objects are located within the object **200**.

[28] The objects **200** may also include constructor and destructor functions known in the art, which can be used to initialize certain object parameters. Constructors can be invoked during object registration or upon the objects **200** first invocation, such as

the initial play of a MP3K file within an MP3 player. Destructors can be invoked during system shutdown or when the playback of a MP3K file is stopped. In addition to constructors and destructors, objects **200** can be invoked by passing messages to the object system message handler. Alternative embodiments of invoking objects **200** may also be used. The object flags (OF) field **230**, within the object header, defines when the object constructors and destructors should be invoked, as illustrated in the following table. The constructor and destructor parameters may also be defined in different locations, such as in a separate section of the object.

OBJECT FLAGS (16) 220															
RES	RES	RES	RES	RES	RES	RES	RES	RES	RES	RES	RES	STP	ULD	CALL	LD
LD	When Set, the constructor of the object will be run when the object is loaded														
CALL	When set, the constructor of the object will be run when the object is first invoked.														
ULD	When set, the destructor of the object will be called when the system shuts down.														
STP	When set, the destructor of the object will be called when the file that initially invoked the object stops.														

[29] Functions referenced by an object **200** can be classified by their functionalities. Classes **250**, the structure of which is illustrated in **FIG. 3**, are used to manage different sets of functions **300**, the structure of which is shown in **FIG. 4**. All of which may be invoked through the member function of an object **200**. The 8 bit class number field **260** and the 8 bit function number field **310** is typically combined to generate a function ID. Each class **250** further includes a number of functions field **270** in order to maintain and track the functions **300** classified thereunder.

[30] Each function **300**, as already mentioned includes a function number **310**. The function **300** also includes a number of parameters field **320** for which a parameter number may be stored such that a parameter **370** may be invoked. Additional fields include function flags (FF) **330**, which should be set, and function types (FT) **340**.

[31] The structure of the parameter **370** is shown in **FIG. 5**. Each parameter **370** is typically defined to include a 8 bit length field **380** and a 32 bit default value field **390**, where parameter information is given and stored.

[32] In one embodiment of the invention, an object **200** and related function data is delivered by a separate file called an attribute file, suffixed by ".fmo" also referred to

herein as an FMO formatted file. Preferably the attribute or FMO formatted file is delivered by concatenating the attribute file at the beginning of a media or MP3 file. However, other delivery methods may include, as mentioned below, providing the attribute file in bulk, within an ID3 tag, or at the end of the media or Mp3 file. The attribute or FMO formatted file is essentially a file that includes encapsulated objects and is essentially comprised of one or more applet objects and one or more corresponding data objects.

[33] **FIG. 10** depicts the composition of the attribute file **1000**. Essentially, FMO formatted files (FMO files) are a transport mechanism for the applet objects **1010** and data objects **1020**. It is important to note that the attribute file need not contain only the applets and data for a single application. Virtually any number of objects **200** can be included within an attribute file **1000** for transport as each object **200** has its own unique 32-bit identifier or group number (GN) **210**. The applet and data objects **1010 and 1020**, respectfully, may contain but are not limited to, object definition, lyrics/text contents, performer descriptions, general data and variables, and multimedia data. As illustrated in **FIG. 10**, there are an "m" number of objects **200_1 - 200_m**. Each object is defined by an applet object **1010_1 - 1010_m** and a data object **1020_1 - 1020_m**.

[34] As previously mentioned, a MP3K file can be generated from a MP3 file by embedding data within the MP3 file prior to encoding the audio bit sequence. **FIG. 6** illustrates the data frame structure **400** for the MP3K data frame **410**. During encoding data or other information as described below may be embedding and synchronized into the MP3K data frames **410**. MP3K bit streams (encoded media sequences) may be composed of MP3K data frames **410**, which can contain a sync word (SW), a group number (GN) **210**, and data chunk(s).

[35] A MP3K data frame **410** may consist of 400 bits which, for a MP3K file formatted with 16-bits of ancillary data **180**, is 25 MP3 frames **110**. The MP3K data frame **410** is defined as 400 bits since the synchronization word will be included once, and the group number **210** will be repeated exactly twice. As mentioned above, the invention provides for MP3 formatted files with 16-bits of reserved ancillary data **180**.

However, the number of ancillary data bits may be completely arbitrary. A physical limit to the minimum number of bits that must be reserved in the ancillary data section **180** of an encoded bit stream will be dependent on the functionality to be implemented and the type of CODEC used as is known in the art.

[36] With specific reference to the MP3K frame **410**, the MP3K frame is divided into 16 sections (data sections) **420**. In each section, one bit of synchronization word **440**, defined as 0xFF00, is embedded. The purpose of the synchronization word **440** is to facilitate location of the beginning of the group number **210**. As further defined herein the group number **210**, which as mentioned above uniquely identifies a specific object **200**, takes 32 bits, which are diversified into four G bits **450**. As such the synchronization word **440** identifies the first G bit **450**, which makes up the 32 bit group number **210**. This is especially critical and difficult when trying to decode a MP3K bit stream in a streaming environment when frames can be dropped. The bit of synchronization word (denoted S) **440** is located in the first bit position of each section **420**. Since the group number (GN) **210** takes 32 bits, the group number is diversified in the data sections **420**. Four bits (denoted G **450**) of the 32 bit group number **450** are stored in each section **420** (one for every five bits except for the first bit). Subsequently, a G bit **450** will be repeated for every eight sections to make up the 32 bit group number **210**. Both S **440** and G **450** bits are allocated in an order of significance, meaning significant bits will be stored first. The spaces marked by x **460** between S **440** and G **450**, or two adjacent G's **450** are used for data storage.

[37] The total space for data storage is then 320 bits, and can be called a data chunk, as is illustrated in **FIG. 7** as **470**. Synchronized data can be coded (data code) by both prefix codes and object dependent codebooks. A prefix code takes two bits and defines code modes (data modes) **480** of the data code, while a codebook specifies object functions. The following table describes one embodiment of prefix code.

Mode	Name	Description
00	NOP	No operation
01	Object	An object
10-11	Reserved	

According to the above table, there are two different code modes 480, "NOP" (00) and "Object" (01). "NOP" tells an MP3K decoder that there is no operation, while "Object" offers some specific information about object functions.

[38] The codebook, associated with a particular MP3K file, is generated based on the content and/or pre-designed objects 200. The objects 200 do not have to be the same for different MP3K files. Therefore, the data code 480 is typically not allowed to cross data chunk 470 borders.

[39] When an object mode is detected (indicated by a 01 data mode), a variable length code containing detailed object information is passed from an MP3K file to the processing device. The information may include, but is not limited to, a number of functions, function indices, and parameter status with values (if any). If a new parameter value (instead of a default value) needs to be specified, the 1-bit parameter status will be set to "1", and a new parameter value will follow, otherwise the parameter status is set to "0". When a functions parameter values are fixed, no status bits or parameter values need to be passed. The code length for number of functions and function indices can be determined from the attribute (FMO formatted) file. After a function index is given, the function's parameter number and the bit length of each parameter can be found from the associated function definition.

[40] **FIG. 8** illustrates one embodiment of object mode code syntax 500. In this embodiment, it is assumed that two functions are involved in a data frame. Function one 510 has two parameters, in which parameter one 530 takes a default value, and parameter two 540 uses a new value 545. Function two 520 has one parameter 550, which uses a new value 555. For this embodiment, it is further assumed that when a new parameter value is specified, it may be only valid for the current MP3K frame. Its default value may not change.

[41] The previously mentioned the attribute file 1000 provides data and other information, which includes object definition, lyrics, text contents, performer descriptions, general data and variables and multimedia data. The attribute or FMO files are comprised of one or more applet objects and the corresponding data objects.

These objects **200** should be managed as to facilitate the compilation of objects based on invocation by other objects and media files for transfer to the processing device. There are essentially three ways the FMO files may be distributed.

[42] The first method is to encapsulate the FMO file within a media file and within an ID3 tag. ID3 and ID3 tag is in reference to the ID3 compression standard. When encapsulating the FMO file within an ID3 tag there is typically only one applet object and the associated data object included in the FMO file.

[43] The second method is to provide the applet and data objects in "bulk." That is to provide a library of objects **200** for downloading into a single FMO file **1000**. These objects may be loaded and paired with the appropriate media file as necessary.

[44] The third method is used when ID3 is not supported; the FMO files are placed at the beginning of a media file, or alternatively at the end. Since the second and third methods are relatively straight forward to individuals skilled in the art, the first method will be discussed in greater detail below.

[45] When encapsulating or embedding the FMO files within the ID3 tag, such as illustrated in **FIG. 11**, the method provides much more functionality for the use of the ID3. The MP3 file **100** includes the ID3 tag **1100**. Within the ID3 tag **1100**, there are provisions for storing private data in an ID3 private frame **1110**. The FMO or attribute file **1000** could thus be stored in the private frame **1110** of the ID3 tag **1100**. The objects **200** contained in the attribute file **1000** are then invoked by embedding invocation data **1120** in the MP3 frames **110**. As mentioned above, the invocation data **1120** could be contained in the MP3K data frames **410** which is embedded and synchronized in the MP3 frames **110** prior to encoding the bit stream.

[46] In greater detail, an ID3 tag comprises several frames. Each frame begins with an ID3 header and is followed by payload data. The structure of the ID3 frame header is illustrated below.

Byte Number	Description
1	Frame identifier. For embedding FMO date, this identifier should be "PRIV" in ASCII (0x50, 0x52, 0x49, 0x56)
2	
3	

4	
5	Size in bytes, most significant byte first
6	
7	Size, least significant byte.
8	
9	Flags, Byte 1
10	Flags, Byte 2

[47] In the first 4 bytes, the ID3 header has provisions for embedding private data within a frame of an ID3 tag. The frame identifier is the character set "PRIV" in the ASCII standard. The frame length (size descriptor), in the second four bytes, can be the length in bytes, of the entire FMO file.

[48] In the ID3 frame header, the size descriptor is followed by two flags bytes with all unused flags cleared. The first byte ("byte 9") can be for status messages, and the second byte ("byte 10") can be for encoding purposes. If an unknown flag is set in the first byte, the frame may not be changed without the bit cleared. If an unknown flag is set in the second byte, it is likely to not be readable.

[49] The following tables illustrate the ID3 flags. The preferred flag settings for the invention are described in the paragraphs following.

ID3 Flags Byte No. 1

7	6	5	4	3	2	1	0
a	b	c	0	0	0	0	0

ID3 Flags Byte No. 2

7	6	5	4	3	2	1	0
i	j	k	0	0	0	0	0

[50] The tag alter preservation flag ("a" for ID3 flags byte no. 1), indicates to the software what should be done with a frame if it is unknown and the tag is altered in any way. This may apply to all kinds of alterations, including, but not limited to, adding more padding and reordering the frames. This bit should always be zero for embedding a FMO file, indicating the frame should be preserved. If the bit was assigned "1" it would indicate the frame should be discarded.

[51] The file alter preservation flag ("b" for ID3 flags byte no. 1), indicates to the software what to do with this frame if it is unknown and the file, excluding the tag is altered. This does not apply when the audio is completely replaced with other audio data. This bit should always be zero for embedding an FMO file; again indicating the file should be preserved and not discarded.

[52] When set, the read only flag ("c" for ID3 flags byte no. 1), tells the software that the contents of this frame is intended to be read only and that changing the contents might break something (e.g. a signature). If the contents are changed, without knowledge in why the frame was flagged read only and without taking the proper means to compensate (e.g. recalculating the signature), the bit should be cleared. All FMO files should be read-only; this bit should be set to one.

[53] The frame compression flag ("i" for ID3 flags byte no. 2), indicates whether the frame is compressed. This bit should be 0 for FMO files meaning frame is not compressed.

[54] The encryption flag ("j" for ID3 flags byte no. 2) indicates whether the frame is encrypted. In one embodiment of the invention, the FMO file has another form of encryption/authentication. Therefore, this bit should be zero indicating the frame is not encrypted.

[55] Last, the grouping identity flag ("k" for ID3 flags byte no.2) indicates whether this frame belongs in a group with other frames. If set, a group identifier byte is added to the object frame header and every frame with the same group identifier belongs to the same group. This bit should always be clear when embedding an FMO file, again to indicate the frame is not encrypted.

[56] In one embodiment of the invention, the first 16-bits of an FMO file **1000** contains the version number of the format included in the FMO file. Each nybble (half a byte) is interpreted as a BCD (binary coded decimal) number. The full version is represented by a number xx.nn, where xx is the upper most significant 16-bits and nn, the lower. Unlike the version number for the FMO file format, the version number for the object **200** can be interpreted as the version of that object **200** only, and not the format. The library software responsible for managing objects **200** may

use this field to purge older objects **200** as needed. The smallest size for any type of data in an FMO file is 8-bits. For larger data sizes, the most significant byte is included first, followed by all lesser significant bytes.

[57] Typically every FMO file **1000** defines more than just one object **200**. This may simplify the distribution and management of these objects. To handle multiple objects **200**, the next word in the FMO file should be interpreted as the number of objects **200** defined within the file. This embodiment does not recognize the value of zero and it should not be used. Further, for each object **200** there can be a 32-bit pointer to that object within the FMO file.

[58] Within the FMO file, all objects begin with an object header. The object header contains information regarding the format, identifier and version of the object **200**. As mentioned above, the object identifier or group number **210** is a unique 32-bit number used to identify the object **200** and is assigned and tracked by a central authority. This method can help ensure trouble-free communication between objects.

[59] A 16-bit version number may be provided in the object header to help identify various versions of objects **200**. One embodiment of the invention may provide a format to be used to insure processing devises interpret the versions number correctly.

[60] The invention further provides for at least one 16-bit word, used to include object flags **230**, that help control how the object is invoked, illustrated in **FIG. 9**. One word **610** of these flags is reserved for use within the object **200** and may be read by a member function of a class **250**, such as CSystem. The class **250** assignments are explained in greater detail below. The invention also provides for an authentication bit **620**, and an encryption bit **630**, with the remaining bits **640** used for future enhancements of the invention.

[61] If the encryption bit E **630** is set, the entire applet object and associated data objects are encrypted. Content providers may then have the burden of distributing encryption keys based on their requirements. For example, the content provided may provide the encryption key directly to an OEM for embedding within their product. For the embodiments embedding encryption keys within the device's firmware, or

stored on a library in a PC, the secrecy of the location and method of archiving of encryption keys may need to be provided.

[62] An additional embodiment of the invention may provide a means for authenticating an object prior to the objects use. That is to say, authentication enables the host to determine whether or not the object is legitimate, un-tampered, and from the owner that is suspected. The authentication bit **620** can be provided whether or not the applet objects and data objects are encrypted. Encryption alone, however, may not ensure authentication. An object can be authenticated as such if the authentication bit **620** is set.

[63] In one embodiment, authentication may be accomplished with RSA public key cryptography. For this embodiment, all processing devices have a copy of the public component of a single master key. The holder of this master key is the Certification Authority (CA) and the master key is referred to as the CA Key. Only the CA has knowledge of the private component of the master key. Content providers who wish to use the authentication mechanism must also have a key. This key is presented to the CA in the form of an X.509 certificate signing request. The CA may then sign this certificate with the CA Key.

[64] In an embodiment of an authenticated FMO stream, the content provider's X.509 certificate (signed by the CA Key) is present. The authenticity of the X.509 may be checked using the processing device copy of the public component of the CA Key. If it is not authentic, it may be ignored.

[65] At the end of each authenticated applet object, a MD5 checksum (computed from both applet and data objects) may be generated by one embodiment, and signed by the content provider's key. To authenticate an object, the signature on the MD5 checksum may be verified using the content providers public key (from the certificate), and the signed checksum can be compared to the signed checksum computed for the object. If the MD5 checksum does not match or the signature is invalid, the entire FMO file may be ignored.

[66] The X.509 certificate length field in ID3 is optional. If the certificate is not included, the length should be set to zero. The X.509 certificate is a certificate for the

public portion of an RSA 1024-bit key in extended X.509 format with a binary encoding. This certificate can be signed by the master key. If the certificate does not prove to be valid, it is ignored. Certificates may usually be less than 1 KBytes in length.

[67] Objects 200 typically have two types of functions 300. That is, functions 300 that can have parameters 370 passed to them and functions 300 that cannot. Member functions that cannot take parameters and do a specific task are called shortcuts. If a function 300 can be defined as a shortcut, its function flag 330 may be set to "1." An object 200 could be constructed using shortcuts or by using member functions, or a combination of both with the only significant difference being the amount of data embedded in the payload. In alternative embodiments, one technique may be more efficient than the others.

[68] Functions 300, regardless of whether it is a member function or shortcut, can be defined as one of the following types, as is set in the Function Type 340 field:

- A single foundation class member function call (Release 1), Type 0 (0X00)
- Multiple foundation class member function calls (Release 2), Type 1 (0x01)
- Interpreted code (Release 3), Type 2 (0x02); and
- Machine dependent code (Release 4), Type 3 (0x03)

[69] The number of member functions defined is specified by an 8-bit value, member function count (MFC). For functions of Type 0, each member function can be defined by the following 5 parameters:

1) MFID: This is the identifier to be embedded within the media file bit stream. Maximally, it is 8-bits in length. However, the actual length (MFIDL) will be defined by the following equation:

$$MFIDL = RND(\log_2(MFC))$$

Where RNDO rounds the result up to the next integer.

2) FF: Function flags as discussed above.
3) FT: Function Type as discussed above.
4) CID: Identifier for the class of the member function.
5) FID: Function identifier of the function to be called.

[70] Following the function identifier (FID), the number of parameters to be defined is included. This number indicates how many parameters will be redefined to be different from the default values of the function. For each parameter, there may be a parameter index (PIDX) indicating which parameter of the function can be redefined followed by the parameter value (PV). The length of PV is determined by the function prototype. However, to simplify decoding, the minimum data chunk for the FMO file can be 8-bits. Therefore, if the parameter is 4-bits, then only the most significant nibble of the byte may be used. The rest will be ignored and should be filled with zeroes.

[71] The MD5 Checksum should be computed for the entire applet object and the associated data objects but should not include the MD5 Checksum or the applet object flags. Each applet object shall have its own MD5 checksum so that an FMO file can be revised to change a single applet object and its associated data objects without affecting the remaining objects. A "RSA Signature" of the MD5 checksum may be required if a certificate is present, otherwise it should be omitted.

[72] Since many applications require the ability to pass data along with the object **200**, the data can be accessed by member functions or shortcuts. Data definitions may be static and as such cannot be changed or may be defined as variables to allow modification during the applet's runtime. Typically, data objects can be handled differently depending on whether they are defined as read only or read/write. Variables that are defined as read only can be kept within the object **200** and not moved elsewhere, saving memory.

[73] There can be at most 256 data or variable objects within an object **200**. The number of data objects defined within an object **200** is determined by the data object count (DOC). Following the DOC, each data object is defined by the following parameters:

- DOID: This is the unique 8-bit identifier of the data object.
- DOF: Data object flags. These flags control how the host handles objects.
- DOT: Data Object type. This determines the type of the data object is.

- DOL: Number of elements in the data object. All variables can be thought of as arrays.

[74] All of the elements in the data object may be defined. The length of a data element is defined by the DOT. However, to simplify object **200** interpreting, the minimum data size may be kept at 8-bits. If a data element is only one bit, the seven least significant bits can be ignored and should be zero. An example of the Data Object Flags (DOF) is shown below:

DATA OBJECT FLAGS (8)							
Res	Res	Res	Res	Res	Res	Res	0 – write/read, 1-read only

[75] All data objects are constructed from a type class. The class identifications (CIDs) and the description identification of each of these classes are shown in the following table:

CID	CLASS NAME (DESCRIPTION)
0x00	TBoolean
0x01	TChar
0x02	TInt
0x03	TLong
0x04	TUChar
0x05	TUInt
0x06	TULong
0x07	Reserved
0x08	KChar
0x09	KCharArray
0x0a	KCharArrayPtr
0x0b	KCharArrayPtrArray
0x0c	CGraphic
0x0d	CInfo
0x0e-0x0f	Reserved
0x10	CSystem
0x11-0x1f	Reserved
0x20	CKaraoke
0x21-0xff	Reserved

The Classes are referred to by a unique 8-bit identifier (CID) rather than by name to minimize the time associated with embedding actual names. However, the Class names listed above (description) could easily be embedded instead of the CIDs.

[76] All type classes have a number of standard functions that an object interpreter uses for accessing the data of the various data objects. Some common functions may include: Create, which creates readable or writtable variables or arrays; GetValue, which is a pointer to an indicated variable, if the variable is an array, an index would be required to access the particular element within the array; SetValue, which enables an application to free up memory for a specified variable, this function also checks the attributes of the variable to determine whether the value can be modified or not; and Remove, which enables an application to free up memory associated with a specified variable.

[77] Still referring to the above table listing the classes, the naming convention for the classes is defined as "T" being a Type; "K" stands for Karaoke, classes define a data type used specifically for the implementation of synchronized data; and "C" stands for Class, standard prefix defining a group of functions/data types that have similar utility.

[78] In further detail, TBoolean defines Boolean variables that can either take on a value of true ('1') or false ('0'). It is also possible to create an array of Boolean values. TChar defines a character that may have a value ranging from -128 to 127 (8 bits). TInt defines an integer value ranging from -16384 to 16383 and are 16-bits in size. TLong defines long variables ranging from -2.15e9 to 2.15e9 and are 32-bits in size. TUChar are unsigned characters with a range of 0 to 255. TUInt are unsigned integers with a range of 0 to 65535. TULong are unsigned long integers with a range from 0 to 4.9e9.

[79] CSystem contains a number of functions used for controlling and gaining information about the host system device (such as the MP3 player, PC or other device). Essentially, this is the interface between an applet object and the system interface. It further enables the ability to create functions that may access peripherals of the host system such as the keyboard, LCD display, or the devices buttons (such as

forward, reverse, play, stop, etc.). For example, the ability to simulate a key press or access files from within an object is easily accomplished by using CSystem member functions.

[80] For example purposes only, some CSystem functions may include HyperLink and SendEmail when the host system device is connected to the Internet. As such, when the function HyperLink is enabled, the device will link to a website, automatically bringing up additional information about the specific object data. For example, if a song is being played, the link could be to an official artist's website, or a website for purchasing lyrics, albums, etc.

[81] CInfo is a collection of functions used to provide rich content of an audio track. The CInfo data structure may take one of five different format types of data: text (which also includes URL or e-mail data), graphical, audio, operational and video. Info Tags may also be included to add additional data to an audio file. The Info Tags may be synchronized within the audio file, placed at the beginning of a track (prior to any music playback) or at the end of a track. The CInfo data structure may also include type fields to format specific information, such as liner notes, Biography information, Label Information, Band Information, Copyright Information, etc. Moreover, the data structure of the CInfo includes flags that determine various properties of the tag, such as whether the tag is: visible at the graphic user interface, read/read-write; static or synchronized within an audio track; containing generic, restricted, commercial or non-commercial information; or containing data that is specifically tailored for one type of host device.

[82] CGraphics is a collection of functions used to decode and display embedded graphics. The format and location of the displayed graphics is controlled through a series of character commands that control the font, color and any formatting information. The following table illustrates one method for controlling the graphical display.

CHARACTER	DESCRIPTION
'\a'	Toggle between normal and command mode
'c'	Clear the current display
'j'	Field upper - left x position
'k'	Field upper - left y position
'l'	Field lower - right x position
'm'	Field lower - right y position
'f'	Font identifier previously associated with a specific font to follow
'e'	The area behind the characters will be erased with a specified color
'g'	Text is stitched in with the current background
'x'	Set cursor to an x position that follows
'y'	Set cursor to a y position that follows
'\v'	Vertical tab
'\r'	Carriage return
'\n'	Vertical tab and carriage return
'v'	Vertically display the characters in the field
'h'	Horizontal display of the characters
't'	Truncate text
'w'	Wrap around the text
'A'	Justify top of left depending upon mode
's'	Justify right or bottom depending upon mode
'd'	Word wrap is desired
'z'	Character wrap is desired
'C'	Color of the text displayed is set to the 24-bit RGB value defined

As such, the graphic can be centered on the screen by indicating the location starting from x00, y00 (left and top portion of the screen). For example in the following: \ahAdx0y0j0k40l40f0\aHello World!\0 the text "Hello World!" will be displayed in a field with an upper-left coordinate of 0,0 and a lower right coordinate of 40,40. The font #0 is used (0 being previously defined as a specific font) and the text will be displayed horizontally starting at location 0,0. In addition word wrap is desired.

[83] CKaraoke class is designed for management and display of synchronized text applications requiring very little information to be embedded within a media bit stream. These are various data objects that are created to completely specify an associated object in the CKaraoke class.

[84] The Karaoke data objects may include KaraokeFlags that contain the ability to control the display and interpretation of the Karaoke object. Other Karaoke data

objects may be associated with the following classes: KCharArray is a listing of phrases, such as the phrases contained in lyrics; KCharArrayPtrArray is an array of pointers which point to the beginning of each verse; HLIndex is a variable indicating which character or word, as determined by the KaraokeFlags, should be highlighted; and VerseNum indicates which verse should be displayed.

[85] When the media file is played, the MP3K data frame **410** references specific objects **200**, either applets or data objects to be invoked. If the data object running is a variable data object, then the user may be able to insert data. For example, if the applet object is running that prompts the user "Would you like additional information?" The user may respond by pressing one of the player's buttons (each MP3 player includes a plurality of buttons, such as forward, reverse, stop, play, and pause, or if the user is player the media file at a pc the applet could assign a key (on the keyboard) for the user to press). The applet may indicate that the user may respond in the affirmative by pressing one of the buttons or keys. When the user enters a response, the applet saves the response by changing the value on a variable data object.

[86] When the user links the player, PC or device to a central database, the central database can download all variable data objects. The variable data objects are further processed to determine if additional information is to be provided to the user. The additional information may be sent from the central database to the user either through a website or through new objects which the player or device will activate accordingly.

[87] This provides the central database with the ability to provide any type of information. Marketing, advertising and product information may now be conducted and displayed during the playing of a media file. For example, if the media file is a song, information such as "Would you like more information regarding this song, group, singer?"; "Would you like to download the lyrics to this song?"; or even "Would you like to download the video to this song?" can be provided to the user. The type and content of the information provided to the user is endless, and the central database could control all processing of the applet and data objects, as well as where

in the MP3 file the controls invoking the applet and data objects are placed. The information could thus be easily synchronized within the MP3 file.

[88] While decoding the media sequence with the an accompanying attribute file, the device medium, with a decoder, will receive an encoded bit stream and will receive the attribute file. During decoding the encoded bit stream is formed into a media sequence containing a plurality of frames packed with segments of the media file and packed with MP3K data. The decoding process will unpack the frames and begin playing the frames as normally. When a frame with a synchronous bit or MP3K frame is unpack the decoder will come across the group number **210** invoking a specific object **200**. The object's **200** functions **300** will be invoked simultaneously and in synchronous play with the media data (unpacked from the frames). If for example the data object contains lyrics, the applet object may run functions that cause the device medium to display the lyrics with the media data and cause the device medium to highlight the lyrics at the appropriate time.

[89] Other objects, as mentioned above, may be invoked to solicit responses from the user, such as advertisements queries or other related and unrelated questions. When answered by the user, the objects will write or save the information on writtable data objects. The next time the user links up to a central database, the information or answers may be accessed by the central database as mentioned above.

[90] Similarly while unpacking the frames, the decoder may invoke Info Tags contained throughout the media file. As mentioned above, the media frames could be embedded with any objects or tags that will facilitate the application of the invention.

[91] The above-described methods and implementation of encoding and decoding media sequences are example methods and implementations. These methods and implementations illustrate one possible approach for encoding and decoding media sequences. The actual implementation may vary from the method discussed. Moreover, various other improvements and modifications to this invention may occur to those skilled in the art, and those improvements and modifications will fall within the scope of this invention as set forth below.

[92] In the implementation of the above methods a system may be implemented for providing a device medium with a media sequence and data synchronized with the media sequence. The system may include any or all of the following, a means for providing a media file, such means may include the ability to download or upload a media file. A means for providing the attribute file containing at least one object, which as mentioned above, each object includes an applet object and a corresponding data object. As mentioned above, the means for providing the attribute file could similarly be the ability to create, download or upload the attribute file. The system should also provide a means for assigning each object a unique identification number (GN). As the media file is segmented and packed into a plurality of frames, a media sequence is created. Embedded within the frames, the system further provides the means to embed the unique identification numbers, such that if the unique identification numbers are embedded at a specific location the objects identified by such unique identification numbers become synchronized to the segment of the media file embedded within the frame. The system should further provide for a means for encoding the media sequence into an encoded bit stream. Finally, the system may also include a means for transferring the media sequence (encoded bit stream) and the attribute file to a device such as the MP3 player, PC, palm pilot, etc. Wherein the device decoding and playing the media sequence will invoke the objects at the same point in the frames containing the segment of media file currently being played.

[93] The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive.

WE CLAIM

1. A method for synchronizing data with a media sequence comprising:

providing at least one object, each object having an applet object and a corresponding data object;

inserting the objects into at least one attribute file;

providing a media sequence, the media sequence being defined as having a plurality of frames containing segments of an original media signal;

embedding at least one synchronous bit into the media sequence, wherein the at least one synchronous bit links to the at least one object; and

encoding the media sequence to form an encoded bit stream.

2. The method of claim 1, wherein the step of embedding the synchronous bit includes embedding the synchronous bit into a frame that synchronizes the object linked to said synchronous bit to the segment of the original signal contained within said frame.

3. The method of claim 2, wherein each frame includes an ancillary field with a predefined reserved space, and the step of embedding the synchronous bit includes embedding the synchronous bit into the predefined reserved spaced of each ancillary field.

4. The method of claim 3 further comprising the step of assigning a unique identifier to each object and tracking each unique identifier.

5. The method of claim 4 further comprising the step of embedding a unique identifier into the ancillary field containing the synchronous bit that links to the object identified by said unique identifier.

6. The method of claim 5, wherein the synchronous bit facilitates the location of the unique identifier within the ancillary field.

7. The method of claims 6 or 1, wherein each object includes a number of functions and said functions may be classified by a number of classes.

8. The method of claim 7, wherein the functions of the data objects may be classified into one of the following classes: TBoolean, TChar, TInt, TLong, TUChar, TUInt, TULong, KChar, KCharArray, KCharArrayPtr, KCharArrayPtrArray, CGraphic, CInfo, CSystem, and CKaraoke.

9. The method of claim 7, wherein a data object may be defined as a read only data object or a read/write data object.

10. The method of claim 1, wherein the media sequence is provided in a format selected from the group of formats consisting of MPEG 1 Layer III, MPEG 2 Layer III, MPEG 2.5 Layer III, and MPEG 2 Layer VII (AAC).

11. The method of claim 1 further comprising the step of attaching the attribute file to the beginning of the media sequence.

12. The method of claim 1 further comprising the step of attaching the attribute file to the end of the media sequence.

13. The method of claim 1 further comprising the step of encapsulating the attribute file within an ID3 tag.

14. The method of claim 3 further comprising the steps of creating an MP3K file as a function of the predefined reserved space of the ancillary fields.

15. The method of claim 14, wherein the MP3K file contains the synchronous bit, the unique identifier and additional embedded data.

16. The method of claim 1 further comprising the step of encrypting the object apart from the step of encoding the media sequence.

17. The method of claim 1 further comprising the step of transferring the encoded bit stream to a device medium.

18. A system for synchronizing data to a media sequence and transferring said synchronized data and media sequence to a device medium, the system comprising:

a means for providing a media file,

a means for providing an attribute file, the attribute file containing at least one object, each object being comprised of at least one applet object, each applet object having a corresponding data object;

a means for concatenating each object to a separate unique link, such that when a unique link is called the object concatenated to said unique link is invoked;

a means for segmenting and packing the media file into a plurality of frames defining a media sequence, wherein each frame may contain a segment of the media file;

a means for embedding a unique link within a frame, of the plurality of frames, at a location that synchronizes the object concatenated to said unique link to the segment of the media file contained within said frame;

a means for encoding the media sequence; and

a means for transferring the media sequence and the attribute file to the device medium, such that when the device medium decodes and plays the media sequence, a object contained in a frame is invoked when the segment of the media file contained in said frame is played back through the device medium.

19. The system of claim 18 wherein each object includes a plurality of functions.

20. The system of claim 19 wherein one of the functions of a data object contains information that is tailored to a specific device medium, such as but not limited to, an portable device player, or a personal computer.

21. The system of claim 19, wherein one of the functions contains information to decode, format and display graphics.

22. The system of claim 19, wherein one of the data objects contains a listing of phrases corresponding to the media file.

23. The system of claim 22, wherein a function of the data object containing the listing of phrases determines when a phrase, of the listing of phrases, should be displayed on the device medium.

24. The system of claim 23, wherein a function of the data object containing the listing of phrases indicates to the device medium when a phrase is to be highlighted.

25. The system of claim 23, wherein when the phrases contains words or characters, a function of the data object containing the listing of phrases, indicates to the device medium when a word or character in a phrases is to be highlighted.

26. The system of claim 19, wherein one of the functions retrieves specific information regarding the device medium such that an applet object corresponding thereto may permit a user to enter information and store said information into a read/write data object corresponding to said applet object.

27. The system of claim 19, wherein one of the functions includes a hyperlink to an Internet address.

28. The system of claim 19, wherein one of the functions permits a user to enter information regarding an E-mail and permits the user to send said E-mail over an internet when the device medium is interconnected to said internet.

29. The system of claim 18, wherein each object may be encrypted.

30. The system of claim 18, wherein each object may be authenticated.

31. The system of claim 18 wherein the media sequence further includes:
an ancillary data field defined within each frame of the media sequence; and
a reserved file being created by reserving a predetermined amount of bit space within a predetermined number of ancillary data fields, such that each media sequence may contain a plurality of reserved files.

32. The system of claim 31, wherein the unique links are embedded within the reserved files.

33. The system of claim 32 further includes embedding a synchronous bit into the first bit position of a reserved file that contains a unique link, the synchronous bit further indicates the location of the unique link within the reserved file.

34. The system of claim 19, wherein one of the functions is defined as an information tag, an information tag contains, but not limited to, one of the following types of data: textual, URL, e-mail, graphical, audio, video or operational.

35. The system of claim 34, wherein the information tag is defined by, but not limited to, one of the following flags: visible/invisible at the user interface, read only/read-write, static/synchronized, generic/restricted, commercial/non-commercial, or specifically tailored for PC/specifically tailored for portable device/specifically tailored for both.

36. The system of claim 18, wherein the media sequence is provided in a format selected from the group of formats consisting of MPEG 1 Layer III, MPEG 2 Layer III, MPEG 2.5 Layer III, and MPEG 2 Layer VII (AAC).

37. The system of claim 18, wherein the attribute file is attached to the beginning of the media sequence.

38. The system of claim 18, wherein the attribute file is attribute file is attached to the end of the media sequence.

39. The system of claim 18, wherein the attribute file is encapsulated within an ID3 tag at the beginning of the media sequence.

40. The system of claim 18, wherein one of the objects contained in the attribute file includes a unique version number of said attribute file to determiner if a second version of said attribute file is available.

41. A method for decoding a media sequence on a device medium comprising:

receiving an encoded bit stream;

receiving an attribute file containing at least one object, each object includes an applet object and a corresponding data object;

decoding the encoded bit stream to form a media sequence containing a plurality of frames packed with segments of a media file and packed with links to invoke the at least one object;

unpacking and playing the plurality of frames, such that when a frame contains a link to invoke the at least one object, invoking the at least one object synchronously with the playing of any segment of media file in said frame.

42. The method of claim 41, further comprising the step of decrypting the object prior to invoking said object.

43. The method of claim 41, further comprising the step of authenticating the object prior to invoking said object.

44. The method of claim 41, wherein the media sequence is provided in a format selected from the group of formats consisting of MPEG 1 Layer III, MPEG 2 Layer III, MPEG 2.5 Layer III, and MPEG 2 Layer VII (AAC).

45. The method of claim 41 further comprising assigning a plurality of functions to each object.

46. The method of claim 45 wherein one of the functions of a data object contains information that is tailored to a specific device medium, such as but not limited to, an portable device player, or a personal computer.

47. The method of claim 45, wherein one of the functions contains information to decode, format and display graphics.

48. The method of claim 45, wherein one of the data objects contains a listing of phrases corresponding to the media file.

49. The method of claim 48, further providing a function associated with the data object to determine when a phrase, of the listing of phrases, should be displayed on the device medium.

50. The method of claim 49, further providing a function associated with the data object to indicate to the device medium when a displayed phrase is to be highlighted.

51. The method of claim 49, further providing a function associated with the data object to indicate to the device medium when a word or character within a displayed phrase is to be highlighted.

52. The method of claim 45, wherein one of the functions retrieves specific information regarding the device medium such that an applet object corresponding thereto may permit a

user to enter information and store said information into a read/write data object corresponding to said applet object.

53. The method of claim 45, wherein one of the functions includes a hyperlink to an Internet address.

54. The method of claim 45, wherein one of the functions permits a user to enter information regarding an E-mail and permits the user to send said E-mail over an internet when the device medium is interconnected to said internet.

55. The method of claim 41, wherein the attribute file is attached to the beginning of the media sequence.

56. The method of claim 41, wherein the attribute file is attribute file is attached to the end of the media sequence.

57. The method of claim 41, wherein the attribute file is encapsulated within an ID3 tag at the beginning of the media sequence.

58. The method of claim 45 further providing a first function that invokes a response from a user operating the device medium.

59. The method of claim 58 further providing a second function that permits the user to utilize specific control of the device medium to respond to said first function.

60. The method of claim 59 further providing a third function that stores a response from the user.

61. The method of claim 60 further providing a fourth function that uploads the response to a central database when the device medium is linked to said central database.

62. A system for decoding a media sequence comprising:

a means for receiving an encoded bit stream;

a means for receiving an attribute file containing at least one object, each object includes an applet object and a corresponding data object;

a means for decoding the encoded bit stream to form a media sequence containing a plurality of frames packed with segments of a media file and packed with links to invoke the at least one object;

a means for unpacking and playing the plurality of frames, such that when a frame contains a link to invoke the at least one object, invoking the at least one object synchronously with the playing of any segment of media file in said frame.

63. The system of claim 62, wherein one of the data objects is a listing of phrases corresponding to the media file.

64. The system of claim 63 further including a function contained in the data object that is an array of pointers, each pointer links to the beginning of each phrase.

65. The system of claim 64, further including a function contained in the data object that indicates to the device medium when a phrase is to be highlighted.

66. The system of claim 64, further including a function contained in the data object that indicates to the device medium when a word or character in a phrases is to be highlighted.

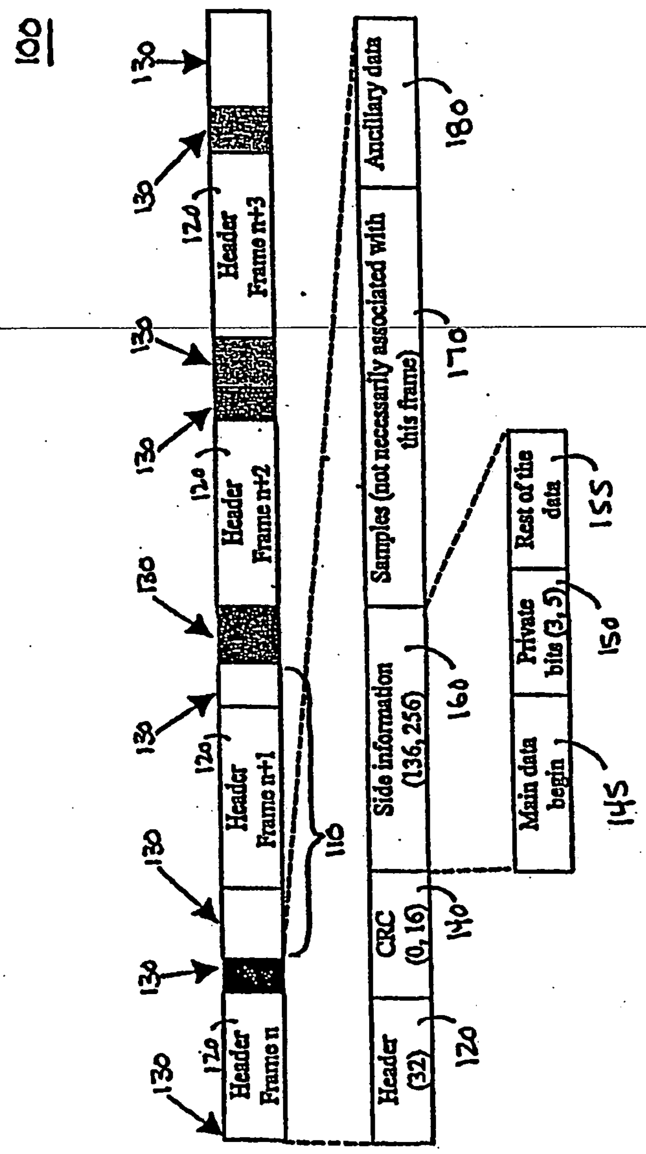

100
130
120
Header Frame n
Header Frame n+1
Header Frame n+2
Header Frame n+3
110
Header (32)
CRC (0, 16)
Side information (136, 256)
Samples (not necessarily associated with this frame)
Ancillary data
120
140
160
170
180
Main data begin
Private bits (3, 5)
Rest of the data
145
150
155

Figure 1/11

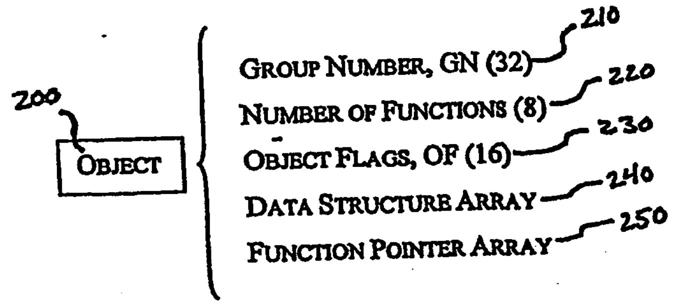


FIGURE 2/11

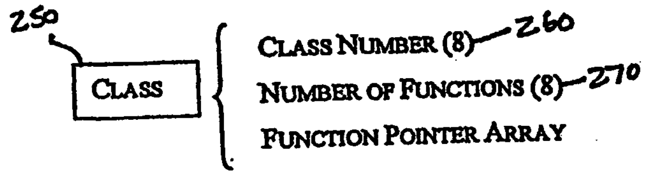


FIGURE 3/11

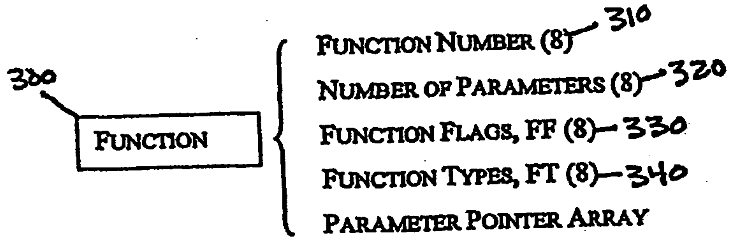


FIGURE 4/11

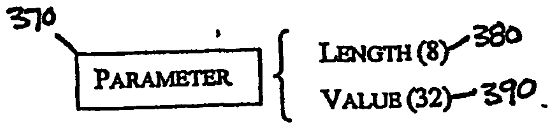


FIGURE 5/11

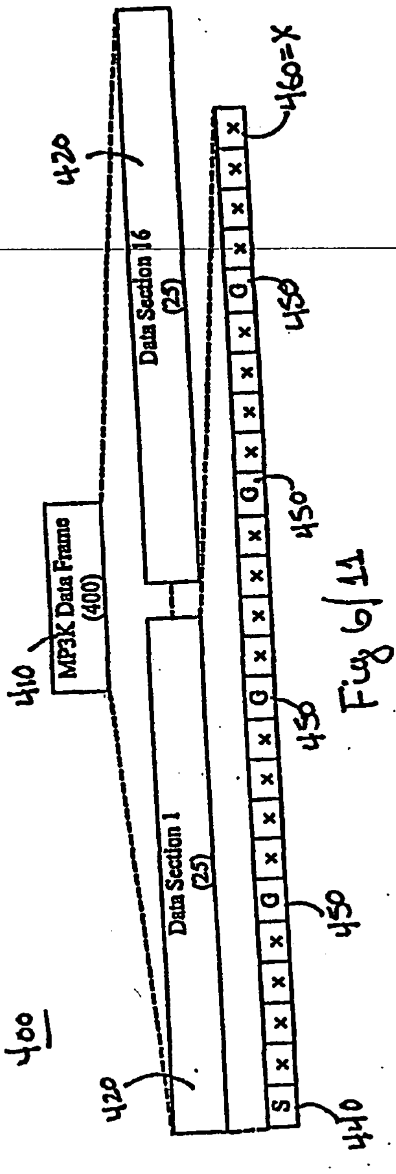


Fig 6/11

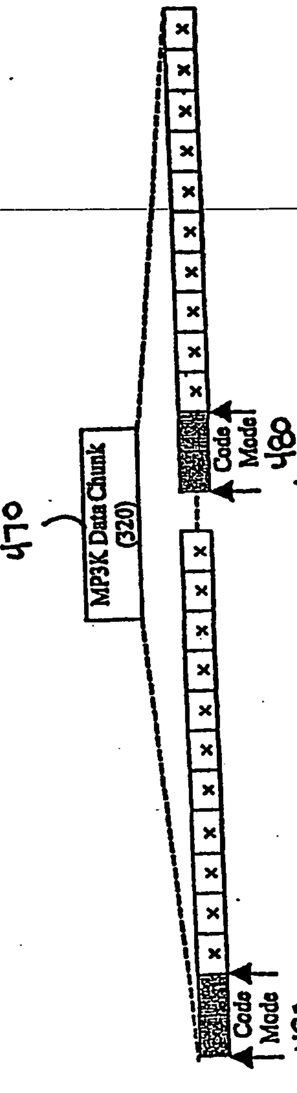


Figure 7/11

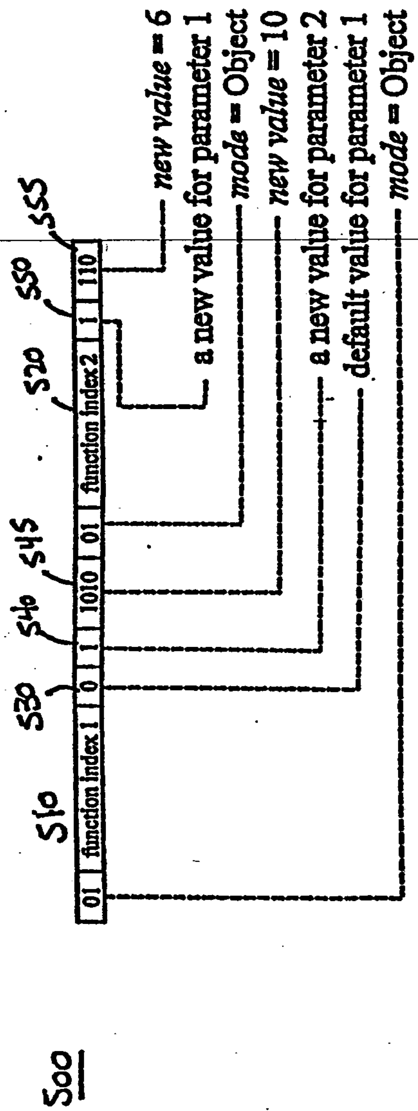


Figure 8|11

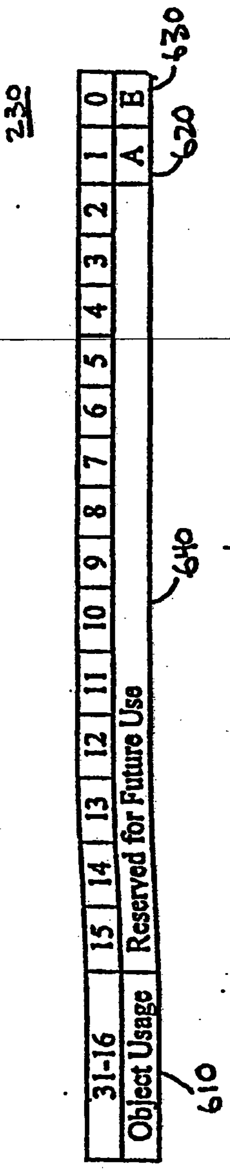
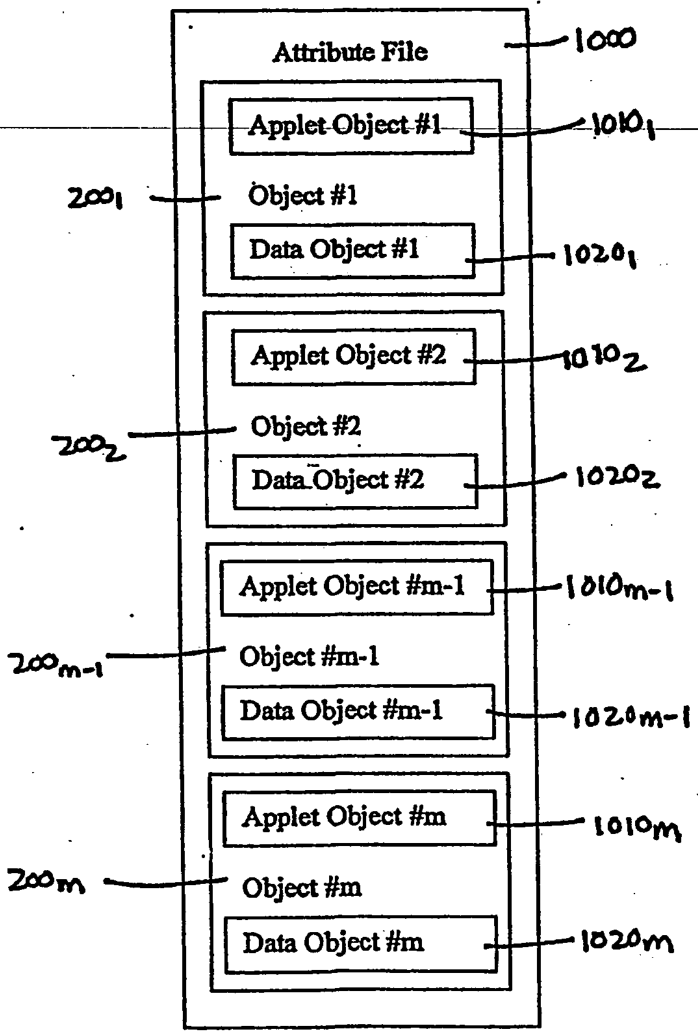


Figure 9|11

FIG. 10 | 11


Attribute File
1000
Applet Object #1
1010_1
200_1
Object #1
Data Object #1
1020_1
Applet Object #2
1010_2
200_2
Object #2
Data Object #2
1020_2
Applet Object #m-1
1010_{m-1}
200_{m-1}
Object #m-1
Data Object #m-1
1020_{m-1}
Applet Object #m
1010_m
200_m
Object #m
Data Object #m
1020_m

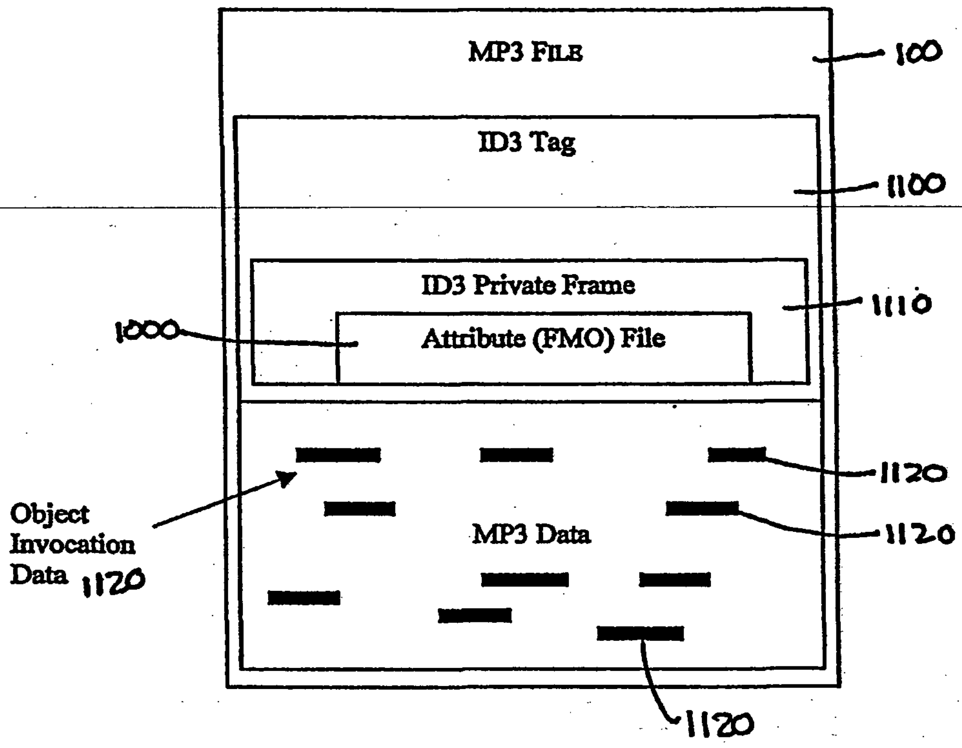

MP3 FILE
100
ID3 Tag
1100
ID3 Private Frame
1110
1000
Attribute (FMO) File
1120
1120
Object Invocation Data 1120
MP3 Data
1120

Figure 11/11

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US02/06710

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : G10L 19/00
US CL : 704/500, 201
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 704/500, 201

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
BRS (US, EPO, JPO, DERWENT)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 98/47084 A (QIAN) 22 October 1998 (22.10.98), entire document	1-66
A	US 5,347,632 A (FILEPP et al.) 13 September 1994 (13.09.94), columns 5, 11-24, 96-98	1-66
A	US 5,959,623 A (VAN HOFF et al.) 28 September 1999 (28.09.99), columns 4-6	1, 18-23, 26-27, 30-, 32, 34-35, 41, 45-46, 53, 62
A	US 6,011,537 A (SLOTZNICK) 04 January 2000 (04.01.2000), columns 7-8, 14-15, 21-25, 30-32	1, 18-21, 41, 45-47, 62

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search 14 June 2002 (14.06.2002)	Date of mailing of the international search report 16 JUL 2002
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703)305-3230	Authorized officer Donald L. Storm [illegible] Telephone No. (703)305-4700

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US02/06710

Continuation of Item 4 of the first sheet:
The title lacks precision. (PCT Rule 4.3) The title has been established to read as follows:

METHOD AND SYSTEM FOR ENCODING AND DECODING SYNCHRONIZED DATA WITHIN A MEDIA SEQUENCE